ARCHITECTURE FOR A UNIVERSAL SERIAL BUS-BASED PC FLASH DISK

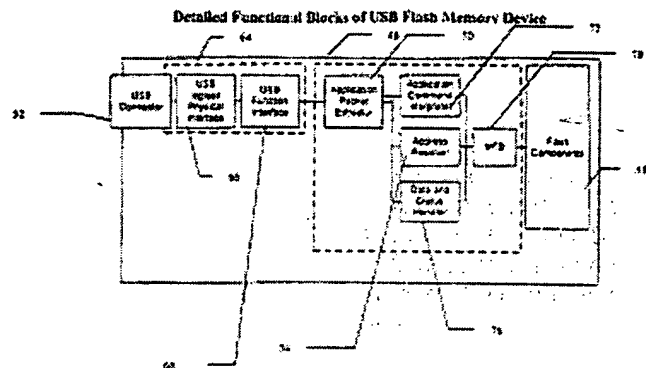|		
Patent number:	CA2334113	
Publication date:	2000-10-12	
Inventor:	BAN AMIR (IL); MORAN DOV (IL); OGDAN ORON (IL)	
Applicant:	MILSYS LTD (IL)	
Classification:		
- international:	G06F13/36	
- european:	G11C7/10L	
Application number:	CA20002334113 20000320	
Priority number(s):	US19990285706 19990405; WO2000US07087 20000320	

Also published as:

- WO0060476 (A1)
- EP1092193 (A1)
- US6148354 (A1)
- EP1092193 (B1)
- AU766478 (B2)

Report a data error here

Abstract of **CA2334113**

A storage unit made of a flash array (58) and a Universal Serial Bus (USB) controller (56) is implemented to be compatible with the USB specification. The unit (46) includes memory modules (58) which can accept write commands a nd read commands from a host (44), and are erasable and non-volatile, referred to as flash modules (58). The USB/flash controller (56) is configured to provid e USB functionality and compatibility along with common flash operations such as programming, reading, and erasing the flash modules (58).

Detailed Functional Blocks of USB Flash Memory Device

Data supplied from the *esp@cenet* database - Worldwide

BEST AVAILABLE COPY

ARCHITECTURE FOR A UNIVERSAL SERIAL BUS-BASED PC FLASH DISK

Description of corresponding document: **US6148354**

FIELD AND BACKGROUND OF THE INVENTION

The present invention is related to semiconductor memory devices, and in particular to erasable and programmable nonvolatile memory modules which are connected to a host platform using the USB PC Bus.

Erasable and programmable non-volatile memory modules, hereinafter referred to as flash memory or flash devices, are known in the art for storage of information. Flash devices include electrically erasable and programmable read-only memories (EEPROMs) made of flash-type, floating-gate transistors and are non-volatile memories similar in functionality and performance to EPROM memories, with an additional functionality that allows an in-circuit, programmable, operation to erase pages of the memory. One example of an implementation of such a flash device is given in U.S. Pat. No. 5,799,168, incorporated by reference as if fully set forth herein.

Flash devices have the advantage of being relatively inexpensive and requiring relatively little power as compared to traditional magnetic storage disks. However, in a flash device, it is not practical to rewrite a previously written area of the memory without a preceding page erase of the area. This limitation of flash devices causes them to be incompatible with typical existing operating system programs, since data cannot be written to an area of memory within the flash device in which data has previously. been written, unless the area is first erased. A software management system, such as that disclosed in U.S. Pat. No. 5,404,485, filed on Mar. 5, 1993, which is incorporated as if fully set forth herein, is required to manage these functions of the flash memory device.

Currently, these flash memory devices have a second limitation, which is that they must be either attached statically to the host platform, or attached and detached dynamically using the PCMCIA [Personal Computer Memory Card International Association] interface. Both implementations have drawbacks, including difficulty of use and high cost.

A more useful implementation would follow the USB standard, as described in the USB Specification Version 1.1 which is incorporated as if fully set forth herein. The USB standard offers a smaller form factor and greater ease of use for the end user, while lowering the cost of the implementation. This standard is specified to be an industry-wide standard promoted by companies such as Compaq Computer Corporation, Microsoft, IBM and Intel to serve as an extension to the PC architecture with a focus on Computer Telephony Integration (CTI), the consumer, and productivity applications.

The criteria which were applied to define the architecture for the USB standard include the ease of PC (personal computer) peripheral expansion, low cost, support of transfer rates up to 12 Mb/second and full support for real-time data, voice, audio, and compressed video. This standard also offers protocol flexibility for mixed-mode isochronous data transfers and asynchronous messaging, integration in commodity device technology and provision of a standard interface for rapid integration into any given host product. In addition, the USB standard represents a single model for cabling and attaching connectors, such that all of the details of the electrical functions, including bus terminations, are isolated from the end user. Through the standard, the peripheral devices are self-identifying, and support automatic mapping of functions to a driver. Furthermore, the standard enables all peripheral devices to be dynamically attachable and re-configurable.

A system constructed according to the USB standard is described by three separate, defined areas: USB interconnection, USB devices and the USB host platform. The USB interconnection is the manner in which USB devices are connected to, and communicate with, the host platform. The associated functions and components include the bus topology, which is the connection model between USB devices and the host platform.

The USB physical interconnection has a tiered star topology. A hub is at the center of each star. Each wire segment is a point-to-point connection between the host platform and a hub or function, or a hub connected to another hub or function.

In terms of a capability stack, the USB tasks which are performed at each layer in the system include a

data flow model and a schedule. A data flow model is the manner in which data moves in the system over the USB between data producers and data consumers. A schedule determines access to the interconnection, which is shared. Such scheduling enables isochronous data transfers to be supported and eliminates arbitration overhead.

The USB itself is a polled bus. The host controller on the host platform initiates all data transfers. All bus transactions involve the transmission of up to three packets. Each transaction begins when the host controller, on a scheduled basis, sends a USB packet describing the type and direction of transaction, the USB device address, and endpoint number. This packet is referred to as the "token packet." The USB device, to which the packet is addressed, selects itself by decoding the appropriate address fields. In a given transaction, data is transferred either from the host platform to a device or from a device to the host platform. The direction of data transfer is specified in the token packet. The source of the transaction then sends a data packet or indicates that the source has no data to transfer. The destination, in general, responds with a handshake packet indicating whether the transfer was successful.

The USB data transfer model between a source and destination on the host platform and an endpoint on a device is referred to as a "pipe". There are two types of pipes: stream and message. Stream data has no USB-defined structure, while message data does. Additionally, pipes have associations of data bandwidth, transfer service type, and endpoint characteristics like directionality and buffer sizes. Most pipes come into existence when a USB device is configured. One message pipe, the default control pipe, always exists once a device is powered, in order to provide access to the configuration, status, and control information for the device.

The transaction schedule for the USB standard permits flow control for some stream pipes. At the hardware level, this prevents situations in which buffers experience underrun or overrun, by using a NAK handshake to throttle the data rate. With the NAK handshake, a transaction is retried when bus time is available. The flow control mechanism permits the construction of flexible schedules which accommodate concurrent servicing of a heterogeneous mix of stream pipes. Thus, multiple stream pipes can be serviced at different intervals with packets of different sizes.

The USB standard, as described, has three main types of packets, including token packets, data packets and handshake packets. An example of each type of packet is shown in background art FIGS. 1-3. Background art FIG. 4 shows an exemplary USB abstract device.

A token packet 10, as shown in background art FIG. 1, features a PID (packet identification) field 12, specifying one of three packet types: IN, OUT, or SETUP. If PID field 12 specifies the IN packet type, the data transaction is defined from a function to the host platform. If PID field 12 specifies the OUT or SETUP packet type, the data transaction is defined from the host platform to a function.

An ADDR field 14 specifies the address, while an ENDP field 16 specifies the endpoint for token packet 10. For OUT and SETUP transactions, in which PID field 12 specifies that token packet 10 is an OUT packet type or a SETUP packet type, ADDR field 14 and ENDP field 16 uniquely identify the endpoint for receiving the subsequent data packet, shown in FIG. 2, which follows after token packet 10. For IN transactions, in which PID field 12 specifies that token packet 10 is an IN packet type, ADDR field 14 and ENDP field 16 uniquely identify which endpoint should transmit a data packet. A CRC5 field 18 contains the checksum, for determining that token packet 10 has been received without corruption. Only host platform can issue token packets 10, such that token packets 10 provide control over transmission of the subsequent data packets.

As shown in background art FIG. 2, a background art USB data packet 20 also features a PID (packet identification) field 22 for identifying the type of data packet. Data packet 20 also features a data field 24 for optionally. containing data, and a CRC field 26 for containing the checksum as previously described.

Background art FIG. 3 shows a background art USB handshake packet 28, which features only a PID (packet identification) field 30. Handshake packets 28 are used to report the status of a data transaction and can return values indicating successful reception of data, command acceptance or rejection, flow control, and halt conditions. Only transaction types which support flow control can return handshake packets 28. Handshake packets 28 are always returned in the handshake phase of a transaction and may be returned, instead of data packets 20, in the data phase of a transaction.

These three different types of packets are exchanged during various phases of the transaction which includes a USB device. A schematic block diagram of the functional blocks in a typical USB device 32 is shown in FIG. 4 for an abstract background art USB device. USB device 32 typically includes a USB electrical interface 34, featuring a cable and a connector, which is a physical interface for receiving and

transmitting electrical signals which are compatible with the USB specification as previously described. The signals are then passed to a logical interface 36, which includes one or more buffers, the device address decoder for decoding the address of the source device for the signals, and a SYNC field synchronizer for synchronizing the signals. Information and structures required for management of USB abstract device 32 as a USB device are stored in a USB class control and enumeration engine 38. A function and device engine 40, also termed the "application", controls and manages the specific functions and properties of USB abstract device 32. In addition, function and device engine 40 also consumes and produces most of the data over the USB bus.

The USB specification does not define the relationship between different entities in USB abstract device 32, however. Rather, the USB specification describes only the requirements for the packets, and for the electrical and physical connection between USB abstract device 32 and the bus. Therefore the connections and relationships shown in background art FIG. 4 are only one example of an implementation which fulfills the requirements of the USB specification. Thus, any specific device for fulfilling the USB specification must have a specifically defined and described architecture.

Unfortunately, no such architecture exists for a flash memory device containing one or more flash memory modules, which would enable the flash memory device to connect to a bus defined according to the USB specification and thereby to form part of a USB system on a host platform. For example, U.S. Pat. No. 5,799,168 does not teach or suggest such an implementation for the flash device. As mentioned previously, such an architecture would be particularly useful for a number of reasons, including low cost, ease of use and transparency to the end user.

There is thus a need for, and it would be useful to have, an architecture for defining and describing a flash memory device which is compatible with a USB system and which follows the USB specification, such that the flash memory device could sit on a USB-defined bus and communicate with the host platform through this bus.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic block diagram of a background art USB token packet structure;

FIG. 2 is a schematic block diagram of a background art USB data packet structure;

FIG. 3 is a schematic block diagram of a background art USB handshake data packet structure;

FIG. 4 is a schematic block diagram of an exemplary background art USB device;

FIG. 5 is a schematic block diagram of a system with a flash USB device functionality according to the present invention;

FIG. 6 is a schematic block diagram of the USB Flash disk;

FIG. 7 is a schematic block diagram of a flash identification request packet;

FIG. 8 is a schematic block diagram of a flash identification status packet;

FIG. 9 is a schematic block diagram of a flash write request packet;

FIG. 10 is a schematic block diagram of a flash write status packet;

FIG. 11 is a schematic block diagram of a flash read request packet;

FIG. 12 is sa schematic block diagram of a flash read status packet;

FIG. 13 is a schematic block diagram of a flash erase request packet; and

FIG. 14 is a schematic block diagram of a flash erase status packet.

SUMMARY OF THE INVENTION

The present invention is of a flash memory device, containing one or more flash modules, in which the flash memory is mapped to the address space of an ASIC or a controller which has a USB-defined electrical interface and a USB-defined logical interface. This controller/ASIC (hereinafter termed a

"controller") supports the USB functionality according to the USB standard, thereby supporting enumeration onto the USB bus, as well as data reception and transmission over USB pipes to and from USB endpoints. This controller also supports the functionality and control of the flash memory device, as well as the processing of command and data packets from the host controller. The host controller uses one of several possible protocols, either standard or proprietary, to signal the next command to be performed to the USB flash controller. Thus, the entire device acts as a dynamically attachable/detachable non-volatile storage device for the host platform.

According to the present invention, there is provided a USB flash memory device for connecting to a USB-defined bus, the flash memory device comprising: (a) at least one flash memory module for storing data; (b) a USB connector for connecting to the USB-defined bus and for sending packets on, and for receiving packets from, the USB-defined bus; and (c) a USB controller for controlling the at least one flash memory module and for controlling the USB connector according to at least one packet received from the USB-defined bus, such that data is written to and read from the at least one flash memory module.

Hereinafter, the term "computer" includes, but is not limited to, personal computers (PC) having an operating system such as DOS, Windows.TM., OS/2.TM. or Linux; Macintosh.TM. computers; computers having JAVA.TM.-OS as the operating system; and graphical workstations such as the computers of Sun Microsystems.TM. and Silicon Graphics.TM., and other computers having some version of the UNIX operating system such as AIX.TM. or SOLARIS.TM. of Sun Microsystems.TM.; or any other known and available operating system, including operating systems such as Windows CE.TM. for embedded systems, including cellular telephones, handheld computational devices and palmtop computational devices, and any other computational device which can be connected to a network. Hereinafter, the term "Windows.TM." includes but is not limited to Windows95.TM., Windows 3.X.TM. in which "x" is an integer such as "1", Windows NT.TM., Windows98.TM., Windows CE.TM. and any upgraded versions of these operating systems by Microsoft Inc. (Seattle, Wash., USA).

DETAILED DESCRIPTION OF THE INVENTION

The present invention is of a flash memory device, containing one or more flash modules, in which the flash memory is mapped to the address space of an ASIC or a controller which has a USB-defined electrical interface and a USB-defined logical interface. This controller/ASIC (hereinafter termed a "controller") supports the USB functionality according to the USB standard, thereby supporting enumeration onto the USB bus, as well as data reception and transmission over USB pipes to and from USB endpoints. This controller also supports the functionality and control of the flash memory device, as well as the processing of command and data packets from the host controller. The host controller uses one of several possible protocols, either standard or proprietary, to signal the next command to be performed to the USB flash controller. Thus, the entire device acts as a dynamically attachable/detachable non-volatile storage device for the host platform.

While the invention is susceptible to various modifications and can be implemented using many alternative forms, the embodiment is shown by way of example in the drawings and will be described in details in the following pages. It should be understood that one of ordinary skill in the art appreciates that the present invention could be implemented in various other ways. The intention is to cover all modifications and alternatives falling within the spirit of the current invention.

The principles and operation of a USB flash device and system according to the present invention may be better understood with reference to the drawings and the accompanying description, it being understood that these drawings are given for illustrative purposes only and are not meant to be limiting.

Referring now to the drawings, FIG. 5 is a schematic block diagram of the main components of a flash memory device and system according to the present invention. A flash memory system 42 includes a host platform 44 as shown. Host platform 44 operates USB flash device 46 as a non-volatile storage space.

Host platform 44 is connected to USB flash device 46 according to the present invention through a USB cable 48. Host platform 44 connects to USB cable 48 through a USB host connector 50, while USB flash device 46 connects to USB cable 48 through a USB flash device connector 52. Host platform 44 features a USB host controller 54 for controlling and managing all USB transfers on the USB bus.

USB flash device 46 features a USB flash device controller 56 for controlling the other components of USB flash device 46 and for providing an interface for USB flash device 46 to the USB bus, USB flash device connector 52 and at least one flash memory module 58. Flash memory module 58 is preferably an array of flash memory modules 58 in which the data is stored.

Whenever USB flash device 46 becomes connected to host platform 44, a standard USB enumeration process takes place. In this process host platform 44 configures USB flash device 46 and the mode of communication with USB flash device 46. Although there are many different methods for configuring USB flash device 46, for the purposes of clarity only and without intending to be limiting, the present invention is explained in greater detail below with regard to a method in which host platform 44 issues commands and requests to USB flash device 46 through one endpoint. Host platform 44 queries USB flash device 46 through the other endpoint for status changes, and receives related packets if any such packets are waiting to be received.

Host platform 44 requests services from USB flash device 46 by sending request packets to USB host controller 54. USB host controller 54 transmits packets on USB cable 48. These requests are received by USB flash device controller 56 when USB flash device 46 is the device on the endpoint of the request. USB flash device controller 56 then performs various operations such as reading, writing or erasing data from or to flash memory module(s) 58, or supporting basic USB functionality such as device enumeration and configuration. USB flash device controller 56 controls flash memory module(s) 58 by using a control line 60 to control the power of flash memory module(s) 58, and also through various other signals such as chip enable, and read and write signals for example. Flash memory module(s) 58 are also connected to USB flash device controller 56 by an address/data bus 62. Address/data bus 62 transfers commands for performing read, write or erase commands on flash memory module(s) 58, as well as the addresses and data for these commands as defined by the manufacturer of flash memory module(s) 58.

In order for USB flash device 46 to notify host platform 44 on the result and status for different operations requested by host platform 44, USB flash device 46 transmits status packets using the "status end point". According to this procedure, host platform 44 checks (polls) for status packets and USB flash device 46 returns either an empty packet if no packets for new status messages are present, or alternatively returns the status packet itself.

A more detailed structure of the functional components of USB flash device 46 is shown in FIG. 6. USB flash device 46 includes the physical and electrical interface defined for the USB standard, shown here as USB flash device connector 52 and a connector interface 64. USB flash device connector 52 receives the electrical signals from USB cable 48 which carries electrical signals from host controller (not shown). These signals are then passed through connector interface 64. Every millisecond, a USB frame is carried on the USB-defined bus, such that packets could be sent to USB flash device 46.

Connector interface 64 then receives these packets through a first interface component, which is a combined physical and logical interface 66. A functional interface 68 is specifically designed to receive token packets as defined in the USB specification and as previously described with regard to FIG. 1. These token packets are related only to particular functional aspects of USB flash device 46 which are required for the USB standard, and do not have any relation to particular application of USB flash device 46 as a flash disk according to the present invention. These token packets and their respective returned data packets enable USB host controller 54 (not shown) and host platform 44 (not shown) to identify USB flash device 46 and allocate resources for USB flash device 46 on the USB bus. Thus, functional interface 68 only supports USB functionality needed for the identification and registration of USB flash device 46 on the USB bus.

USB flash device 46 also features an application packet extractor 70 which extracts the application data and commands from the USB application packets, such that application packet extractor 70 supports only application related packets. Next, any requests to USB flash device 46 by host platform 44 (not shown), in the form of read, write, identify and erase commands, are interpreted by an application command interpreter 72. For any commands which involve data or an address, such as read, write and erase commands, an address resolve module 74 translates the address from the logical address space to the physical address space. Host platform 44 (not shown) relates to a linear address space of logical addresses, while USB flash device 46 contains at least one, and preferably a plurality of, flash modules 58, each of which has a physical address space. Thus, a translation must be performed between the logical address space of host platform 44 (not shown) and physical address space or spaces of USB flash device 46. There are many ways to implement such a translation which are suitable for the present invention. One example of a suitable implementation of an address translation method is described with regard to U.S. Pat. No. 5,404,485, previously incorporated by reference as if fully set forth herein, which teaches a method for managing a flash memory as a flash disk and which is suitable for operation with the present invention.

A data handler 76 handles data related aspects of any received commands, and conveying the data through functional interface 68 to and from flash module(s) 58. Optionally and preferably, data handler 76 performs any error correction and detection methods. Application command interpreter 72, data handler

76 and address resolve module 74 all operate with an underlying Memory Technology Driver (MTD) 78 to write, read or erase a particular flash module 58 and the desired address on that flash module 58.

Host platform 44 checks for status changes in USB flash device 46 and reads status packets from USB flash device 46 when a new status packet is available. Using these status packets, USB flash device 46 can transmit, to host platform 44, the results of different commands issued by host platform 44 in its requests (not shown). For example, the read command status packet contains one of the available status words such as "success", "error" or "invalid address", which enables host-platform 44 to determine the result of the read command (not shown). Similarly, the erase status packet contains a status word indicating the completion of the erase process. A write status packet is used by USB flash device 46 to notify host platform 44 about the result of the write command, for example whether the command was successful or erroneous, and whether USB flash device 46 is ready for additional write requests from host platform 44.

A Memory Technology Driver, or MTD 78 typically contains routines to read, write and erase the flash memory device controlled by the controller operating MTD 78. In addition, MTD 78 optionally contains an identification routine for recognizing the proper type of flash memory device for which MTD 78 was designed, so that the controller can determine which MTD should be activated upon interacting with a particular flash memory device array. In addition, an identification routine should be able to detect the size of the array of flash memory devices, including the number of flash memory devices within the array, and various features of the flash array geometry, such as interleaving and bus width. This information later enables host platform 44 platform to determine the address space and size of the storage media. U.S. Pat. No. 5,799,168, previously incorporated by reference, discloses an example of such an MTD for a flash device.

Using the above described protocol and architecture, host platform 44 can optionally implement any application which is implementable with any regular memory mapped or I/O mapped flash memory device. For example, host platform 44 can give a standard block device interface to each application, such as a magnetic storage medium "hard disk" drive, as disclosed in the previously described U.S. Pat. No. 5,404,485.

As an example of a preferred embodiment of the present invention, the operation of a host system connected to a USB flash device according to the present invention is described with regard to the processes of identifying, programming, reading and erasing the flash device. For the purposes of illustration only and without intending to be limiting in any way, the exemplary USB flash device has an array of two flash memory modules, each of which is 64 Mbit in size. The address translation table is within the flash device so that host platform operates with logical addresses. All commands and return codes between the flash device and the host platform are carried on USB data packets, and are transferred through USB data pipes. The exact structure of the packets, pipes and timings are described in the USB specification.

The operation of the exemplary device and system according to the present invention is as follows. When the USB flash device is first connected to the host platform, the USB host controller assigns an address to the USB flash device on the USB bus, and also assigns resources as described in the USB specification. The USB flash device actually asks the host platform to assign these resources, and must inform the host platform how much of these resources are needed. Thus, the USB flash disk can optionally support slower device speeds if the USB host platform has already allocated resources to other devices.

The USB controller also negotiates with the flash modules and determines the size and manufacturing type of these modules. The controller then builds an identification structure holding this information, as well as the translation table and logical address space.

After the USB host controller identifies the USB flash device, the host platform often uploads a USB client driver. The driver issues an identification request command to the USB host controller, causing the controller to transmit an identification data packet 80, shown in FIG. 7. Identification packet 80 contains PID field 22 and checksum field 26, as described previously for background art FIG. 2. Identification packet 80 also contains an "identify" operation code in an operation code field 82. The packet extractor of the USB flash device receives identification data packet 80 and transfers the operating code of the "identify" command to the application command interpreter.

In response to the "identify" command, the flash device then sends an identification data packet 84, shown in FIG. 8. In addition to the fields shown in FIG. 7, identification data packet 84 also contains information about the size of the flash device in a flash device size field 86, as well as information about the size of the minimal erase unit for erasing the flash memory in an erase unit size field 88.

All of the packets described in this example are only data packets which are sent on the USB bus. Before each data packet is sent, a USB token packet is transmitted, instructing the USB controller as to the identity of the device end point to which the data packet should be transmitted. Upon successful reception of the packet, the USB controller issues a USB ACK packet as described in the USB specification.

Once the device drivers in the host platform receive this status packet, the drivers can start issuing read and write commands to the USB flash device with the application commands. When a write request is sent, a USB data packet with the operation code for the "write" command, and the buffer containing the data, is transferred to the USB flash device. A write data packet 90 is shown in FIG. 9, which again includes the fields shown previously in FIG. 8, except that write data packet 90 also includes a write field 92 with the "write" operational code; an ADDR field 94 with the logical address to be written; a LEN field 96 with the length to be written; and a DATA field 98 which contains the actual data to write. The packet extractor extracts the operational code from write data packet 90 and transfers this code to the application command interpreter. The logical address is transferred to the address resolve module which translates this logical address to a physical address on one of the flash modules. The data handler optionally calculates error correction and detection mechanisms if employed by the USB flash device. Once all of the flash memory modules are ready, a "write" command is sent to the flash module or modules containing the physical address, which may optionally span across more than one flash module to the MTD block. The MTD block then issues a "write" command on the data/address bus which connects the flash modules to the USB device controller. Once the operation is complete and a status packet is returned to the MTD, the result of the operation is transmitted to the host controller and passed to the device driver in the host platform.

When the flash controller finishes the writing process, the controller signals to the host platform that the status of the USB flash memory device has changed, by sending a "write status" packet 100, as shown in FIG. 10. In place of data field 98, write status packet 100 contains a status field 102. The host platform reads the status packets from the flash memory device, and from write status packet 100, the host platform retrieves information on the completion status of the write command by reading status field 102. In this example, the flash memory device repeats ADDR field 94 and LEN field 96 in order for the host platform to have a reference to the specific command related to status packet 100.

As shown in FIG. 11, a "read request" packet 104 contains the operation code for the "read" command in a read field 106, and the logical address of the desired location from which the flash controller should read in an ADDR field 108. Upon receiving this command, the flash controller issues a read command to the MTD block, after the address resolve module has translated the address contained in ADDR field 108 to a specific physical address in one of the flash components.

When the flash controller receives the data from the flash device, either after the read command was issued, or if an error occurred, the flash controller sends a signal to the host platform to indicate that a new status packet must be read. The host platform issues a read request and receives a "read status" packet 110 as shown in FIG. 12. Read status packet 110 contains the address of the read data in ADDR field 108, as well as the length of the read data in a LEN field 112 and the data itself in a data field 114. Read status packet 110 also features the status word, according to which the operation was completed, in a status field 116. The read operation can be completed with many different status situations such as success, fail, error detected, invalid address, invalid length and so forth.

When the host platform needs to erase an erase unit in the flash device, the host platform issues an "erase request" packet 118, shown in FIG. 13. This packet contains the "erase" operation code in an erase field 120, and the logical address of the erase unit in an ADDR field 122. Upon receiving such a request, the flash controller translates the logical address to a physical erase unit address on one of the physical address spaces of the flash modules, and issues an erase command to the MTD block.

The erase process generally takes more time then a read or write process. When this erase process is finished, the controller notifies the host platform a new status packet is ready to transmit. The controller then transmits an "erase status" packet 124, as shown in FIG. 14. Erase status packet 124 contains the address of the erased unit in ADDR field 122, thereby providing the host platform with a reference to the erase requests. The status according to which the operation was completed is provided in a status field 126.

It will be appreciated that the above descriptions are intended only to serve as examples, and that many other embodiments are possible within the spirit and the scope of the present invention.

ARCHITECTURE FOR A UNIVERSAL SERIAL BUS-BASED PC FLASH DISK

Claims of corresponding document: **US6148354**

What is claimed is:

1. A USB flash memory device for connecting to a USB-defined bus, the flash memory device comprising:
(a) at least one flash memory module for storing data;
(b) a USB connector for connecting to the USB-defined bus and for sending packets on, and for receiving packets from, the USB-defined bus;
(c) a USB controller for controlling said at least one flash memory module and for controlling said USB connector according to at least one packet received from the USB-defined bus, such that data is written to and read from said at least one flash memory module;
(d) an electrical interface for connecting to said USB connector and for receiving said packets from said USB connector as a plurality of electrical signals;
(e) a logical interface for connecting to said electrical interface and for translating said plurality of electrical signals to logic signals, said logic signals being passed to said at least one flash memory module;
(f) a functional interface for receiving said logic signals such that if said logic signals represent a USB functional packet, said functional interface sends a USB command to said USB controller according to said USB functional packet;
(g) an application packet extractor for connecting to said logical interface and for receiving said logic signals, said application packet extractor extracting at least one packet from said logic signals; and
(h) an application command interpreter for receiving said at least one packet and for determining a command according to said at least one packet, said command being passed to said USB controller.

2. The flash memory device of claim 1, further comprising:
(i) an address resolver module for receiving said at least one packet and for resolving an address contained in said at least one packet, said address being sent to said USB controller, such that said command is performed according to said address.

3. The flash memory device of claim 2, wherein said command is a write command for writing data to said at least one flash memory module and said address is a logical address for writing said data, such that said address resolver module resolves said logical address to a physical address of said at least one flash memory module.

4. The flash memory device of claim 2, wherein said command is a read command for reading data from said a least one flash memory module and said address is a logical address for reading said data, such that said address resolver module resolves said logical address to a physical address of said at least one flash memory module.

5. The flash memory device of claim 2, further comprising:
(j) a data handler for performing an error detection and correction routine for said at least one flash memory module.

6. The flash memory device of claim 5, further comprising:
(k) a status handler for receiving said USB functional packet from said functional interface, and for sending a status packet concerning a status of said at least one flash memory module according to said USB functional packet.

7. The flash memory device of claim 6, further comprising:
(l) a MTD (memory technology driver) for receiving a write command and physical address of said at least one flash memory module, and for performing said write command to said physical address.